AUSTRALIAN UNIX USERS GROUP NEWSLETTER

UNIX USER GROUP MEETING

The University of Wollongong was the venue last month for a successful meeting of the User Group with about 30 persons in attendance. For the benefit of those unable to attend a brief summary of the days talks and events appears on page 2, the summary was produced from my rough notes and I do hope I have not misquoted or misrepresented what was said.

It was decided that the next User Group meeting is to be held at the AGSM in the University of New South Wales in August this year. However at the 'time of going to press' it appears that Dennis Ritchie will be in Sydney in early September for the "Symposium on Langauge Design and Programming Methodology". If this eventuates then I will attempt to arrange the next meeting so that Dennis might attend.

The IFIP conference in Melbourne in October 1980 will potentially provide the possibility for a User Group Meeting with overseas participation. MELBOURNE will investigate and proceed if feasible.

LEVEL 7

Reliable sources indicate that the release of level 7 is imminent. I know we've all heard that before but apparently prices have been set - commercial licences will be even more expensive. There will be two distinct licences for level 7 one for PDP11s and another for VAX 11/780s.

CONTRIBUTIONS

All contributions to the newsletter will be greatly appreciated.

Ian Johnstone
AGSM
PO Box 1
Kensington 2033
AUSTRALIA

(02) 662-3752

SIXTH AUSTRALIAN UNIX MEETING

Friday, March 16, 1979, Pentagon 5, University of Wollongong.

PROGRAMME

9:30-10:00 a.m.	Morning Tea, Northern Lounge Union Building.
10:00 a.m.	Report on visit to Bell Laboratories, Juris REINFELDS, University of Wollongong.
10:15 a.m.	PWB/UNIX, Ian JOHNSTONE, University of New South Wales.
10:30 a.m.	PWB - Memorandum Macros, Peter IVANOV, University of New South Wales.
10:45 a.m.	Writing Portable C-Programs, Richard MILLER, University of Wollongong.
11:15 a.m.	10 minute break
11:30 a.m.	Interprocess Communication for a Distributed System, Doug RYAN, CSIRO Canberra.
11:45 a.m.	Tektronix Graphics Package (in C), Richard GREVIS, University of New South Wales.
12:00 noon	8085 Assembler and Emulator (in C), Adrian FREED, University of New South Wales.
12:15 p.m.	Relocatable Assemblers and Common Link Loader for M6800, I8080, Z80 Microprocessors, Richard KELLY, University of Queensland.
12:30 p.m.	A General Cross-Assembler for Microcomputers, Ross NEALON, University of Wollongong.
12:45 p.m.	DED - Display Editor, Jeff TOBIAS, AAEC, Lucas Heights.
1:00-2:00 p.m.	lunch
2:00-4:00 p.m.	Business meeting and discussions on the following topics: - state of Interdata UNIX - summary from each site on activities and interests - "restricted" shells for undergraduate courses - software available under UNIX (compilers, DBMS, etc.) - who can teach about databases in universities? - teletype handlers - interactive graphics and the graphics standard - UNIX portability (IBM 370 architecture).

MARCH 16 1979 - MEETING NOTES

The following were present:

Geoff Dengate	Prentice Computer Centre QUEENSLAND
Chris Doney	Computer Services Unit UNSW
Robert Elz	Computer Science MELBOURNE
Adrian Freed	AGSM UNSW
George Gerrity	Royal Military College DUNTROON
Richard Grevis	AGSM UNSW
Wayne Harris	University of Sydney
Marc Hillman	University of Sydney
John Holden	University of Sydney
Dave Horsfall	Computer Services Unit UNSW
Andrew Hume	AGSM UNSW
Ian Jackson	Basser Computer Science SYDNEY
Ian Johnstone	AGSM UNSW
Dick Kelly	Computer Science QUEENSLAND
Bob Kummerfield	Basser Computer Science SYDNEY
Piers Lauder	Basser Computer Science SYDNEY
Chris Maltby	Computer Science UNSW
Ken McDonell	Computer Science MELBOURNE
Craig McGregor	Architecture UNSW
Richard Miller	Computer Science WOLLONGONG
Ross Nealon	Computer Science WOLLONGONG
Glynn Peady	Australian Atomic Energy Commission
Juris Reinfelds	Computer Science WOLLONGONG
Brian Rouswell	Computing Centre SYDNEY
Chris Rowles	Chemical Engineering SYDNEY
Doug Ryan	CSIRO Canberra
Ken Schofield	Commerce UNSW
Jeff Tobias	Australian Atomic Energy Commission
Colin Webb	Computer Services Unit UNSW

1 RECENT TRIP TO USA juris reinfelds

Release 7 ready except for the lawyers. Murray Hill is big - 3000 scientists wotking in groups of ten at most - all doing research that is somehow connected to telephones!! Three groups look after UNIX: Research, PWB and the 'maintainers'. At BELL they are working very hard on Software Tools. VAX/UNIX uses 90Kb (45Kb text) all native mode - no demand paging. Steve Bourne working on ALGOL68 for UNIX but it won't fit. Ritchie is working on C optimization. Thomson is still playing chess. McIlroy working on speech synthesis, Latin and English in particular. Aho & Ullman now working with databases. Lesk is investigating inverted indices.

2 PWB/UNIX ian johnstone

The Programmers Work Bench (PWB) version of UNIX is definitely worth having - it costs the same as other UNIX licenses. Useful programs such as MAKE and SCCS exist for maintaining source in usable form

3 MEMORANDUM MACROS andrew hume

Peter Ivanov sent his apologies for not attending. Andrew volunteered to give an impromptu presentation. Memorandum Macros (MM) from PWB are not the ultimate in macro packages but they do provide a means for ordinary mortals to obtain professional results from NROFF (TROFF).

Portable C programs - Can it be done ? YES. Is it worth doing ? YES. What is the problem ? C semantics are not well defined mainly for efficiency - rigid definition leads to problems.

Portability, between operating systems, computers and different versions of C, of data and programs are all worthy of serious consideration.

Size of objects between computers is a real problem. The most obvious differences are listed below. Which is normal ? Which is abnormal ?

	PDP-11		INTERDATA	
	size	align	size	align
char	1	1	1	1
short	2	2	2	2
int/ptr	2	2	4	4
long	4	2	4	4
float	4	2	4	4
double	8	2	8	8
struct or array	2*n	2	4*n	8

More obscure problems result from the following differnces

PDP-11	INTERDATA
char is signed	char is unsigned
integers stored with low byte first	integers stored with high byte first
>> is arithmetic	>> is logical
bit fields address from right to left	bit fields address from left to right
multiple extern definitions are alllowed	multiple extern definitions are not allowed

MORE BAD NEWS

PDP11/UNIX supports dynamic stack growth whereas the INTERDATA version does not. ASCII is not universal. The size of a pointer is not necessarily the size of an integer. Order of evaluation especially in function calls. Multicharacter constants. Case and length of external names. Negative Integer Division.

EXAMPLES - the code you are about to see is true .... Only the names have been changed to protect the innocent.

1.

```
putchar( c )                        | what type is c ? what size ?
{
        write(1,&c,1);              | which byte on which machine ?
}
```

correctly written:

```
putchar( c )
{
        char ch;
        ch = c;                     | really a char
        write(1,&c,1);
}
```

2.

```
getw()                          | function to return next word
{
        int w;
        read(0,&w,2);           | are integers always two bytes ??
}                               | use 'sizeof w' not 2
```

3.

```
getw()                          | function to return next word
{
        int a,b;
        a = getchar();
        b = getchar();
        return (b<<8)|a;        | two chars per int ??
}
```

no easy fix - various messy solutions were demonstrated

4.

```
long bug()
{
        long lret;
        int a,b;
        struct { int word[]; };
        .
        .
        lret.word[0] = a;
        lret.word[1] = b;
        return lret;            | YUK
}
```

5.

```
rounding to even value
        int x;
use
        x =& ~01
not
        x =& 0177776            | is x 2 or 4 bytes long ?
if really want to round to word boundary
then
        x =& ~(sizeof x-1);
```

TOOLS FOR PORTABILITY

GENERALIZATION: use sizeof, union and cast. PARAMETERIZATION: use #define and typedef. ISOLATION: use #include to keep instalation specific items together. CONDITIONAL COMPILATION: use '#ifdef UNIX' and '#ifdef PDP11' (if it existed). DOCUMENTATION: use it. /* warning lark's vomit */ PROGRAM TOOLS: lint is useful.

5 INTERPROCESSOR COMMUNICATION FOR A DUSTRIBUTED NETWORK doug ryan

wanted a distributed data base system via CSIRONET - needed initially good interprocess communication for successful network UNIX. UNIX had pipes but these only useful if common parent has set up the pipe. Two aspects: message passing and synchronization. Tried to implement message passing as just another i/o device within the system - ie no need to know it was one of these devices, but in some cases want to have control over communication. Got message switching going.

6 TEKTONICS GRAPHICS PACKAGE richard grevis

A library of C subroutines was described which are capable of fully utilizing Tektonix 4006, 4010 and 4662 flatbed plotter.

7 8085 ASSEMBLER AND EMULATOR adrian freed

Non-relocating assembler for the 8085 (and 8080) was described, along with an emulator which was used to test programs. This system, together with the surrounding UNIX environment, is a vast improvement on the normal (expensive) commercial development systems.

8 RELOCATABLE ASSEMBLERS richard kelly

Relocatable assemblers and a common link loader for the 6800, I8080 and Z80 microprocessors. Assemblers have a common syntax but also allow manufacturers syntax to be used. There are libraries of routines available for all three machines, and an interactive debugger common to all.

9 A GENERAL CROSS-ASSEMBLER ross nealon

Table driven cross-assembler for microprocessors was described. The assembler is two pass and is written in C. It reads a file describing the microprocessor, including assembler syntax, and produces a standard object file. Tables exist for 6800, 6502 and 8086 uPs. New tables can be created with minimal effort, half day for most with some stretching to a day.

10 DED jeff tobias

DED (Display EDitor) is an exciting editor for very fast display terminals with cursor addressing. Its main feature is that 'what you see is what you get' - in other words the section of the file currently being edited is displayed on the screen and it can be manipulated by various control keys so that the results of all editing is obvious. It has the ability to do all normal editing. One feature is a buffer to hold an editor command which can then itself be edited. It performs best with a modified "tty.c", however it will work with the standard driver.

11 MISCELANIA

WOLLONGONG have C-compilers for 6502 and 8080. INTERDATA/UNIX as distributed by the University of Wollongong is now in use at 9 sites and costs $200 for academic license and $2000 for commercial providing of course a current BELL licence for UNIX is also held. BELL interdata/unix only runs on 832 whereas wollongong version runs on 732 and 832.

Chris Rowles reported that MINI-UNIX was now alive and well... supports all things .... UNFORTUNATELY now that he has got it all going they are getting a 11/60. Mini-UNIX documentation leaves a lot to be desired. He believes mini-UNIX could be simply modified to run on an LSI-11 given LSX/UNIX is not to be released by Bell.

Dick Kelly reported the availability of a plugin card to turn an ADM-3A into a 4010 emulator.

Jeff Tobias has a modification of the University of York Modula compiler that produces 8086 code.

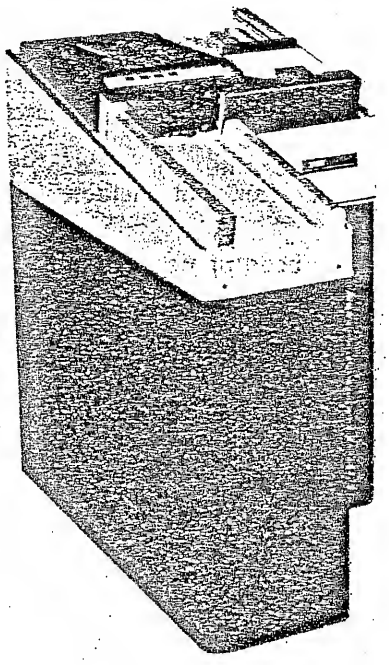
During the discussion of DED, the question of teletype drivers was brought up. This in turn raised the question of delete and erase characters and whether these should be standardised and if so how? Correspondence on this matter is eagerly sought.

the 1978-79 DEC peripherals handbook contains the following page

CHAPTER 5

LINE PRINTERS

LP11/LS11/LA11



← A cardreader or is it a lineprinter in disguise ??

The LP11/LS11/LA11 line printer systems are high-speed printer systems designed to interface with the PDP-11 UNIBUS family of processors.

The **LP11** systems consist of two major components, a line printer and an interface unit referred to as the LP11 controller.

The **LS11** systems consist of two components, a line printer and an interface module, referred to as the LS11 controller.

The **LA11** systems also consist of two components, a DIGITAL line printer, LA180, and a LA11 controller.

LP11 LINE PRINTER SYSTEMS

The LP11 Line Printer system is designed to operate on-line with the PDP-11 UNIBUS systems and associated peripherals such as paper tape readers, magnetic tape units, card readers, or communications terminals. The line printer is mounted in a free-stand-

A special for those who prefer hardcopy !!

Believed to make CPUs redundant!

1058

Bell Laboratories **Cover Sheet for Technical Memorandum**

The information contained herein is for the use of employees of Bell Laboratories and is not for publication. (See GEI 13.9-3)

Title- **C-Language Oriented Microprogram for the HP-21MX**

Date- **February 27, 1975**

TM- **75-1273-1**

Other Keywords-

Author	Location	Extension	Charging Case- 39199
A. G. Fraser	**MH 2C520**	**3685**	Filing Case- 39199-11
D. M. Ritchie	**MH 2C517**	**3770**	

ABSTRACT

The UNIX time-sharing system is presently implemented on the DEC PDP11/40 and PDP11/45 computers. Since the operating system itself and much of its software is written in the C language, an implementation of a UNIX-based system on another computer would be much easier, and much more efficient, if the new machine were particularly congenial to the requirements of C. Drawing on a study of the code generated for a sample of PDP11 C programs, this paper proposes an instruction-set architecture designed for implementation on the Hewlett-Packard 21MX micro-programmed processor. The design, which includes the crucial parts of the actual micro-program, is detailed enough to determine that the resulting machine would be a bit slower than that of the PDP11/40, and the size of compiled C programs would be smaller.

Pages Text 13	Other 0	Total 13	
No. Figures 0	No. Tables 3	No. Refs. 3	

SEE REVERSE SIDE FOR DISTRIBUTION LIST

Bell Laboratories

Subject: C-Language Oriented Microprogram for the HP-21MX
Case- 39199 -- File- 39199-11

date: February 27, 1975

from: A. G. Fraser
D. M. Ritchie

TM: 75-1273-1

MEMORANDUM FOR FILE

Introduction.

It is currently both practical and economically attractive to construct a general-purpose computer by micro-programming a simpler special purpose machine. The time may soon come when a corporation such as Bell Laboratories can standardize upon a machine/language design and yet have the option of buying the hardware from several sources. The limited extent to which that might now be possible is illustrated by an investigation into using Hewlett-Packard hardware in place of the Digital Equipment machine on which UNIX software normally runs.

The UNIX operating system[1] currently runs on PDP11/40 and PDP11/45 computers. Most of the operating system and applications software are written in the language C [2]. The language encourages but does not attempt to enforce the portability of programs between machines. Recent experience in moving fairly substantial UNIX programs, written in C, to the IBM 370 and the HIS 6070 suggests that most of the effort of transporting programs lies not in the code itself but in the interface to the operating system, in spite of the fact that the hardware environments of the three machines involved are quite diverse. Nevertheless, for efficiency reasons, the following architectural features should be provided.

a) UNIX strategy relies heavily upon multi-programming and so address mapping hardware must be available to allow several processes to occupy memory simultaneously.

b) Byte variables, pointers to bytes and arithmetic on byte pointers are heavily used and should be provided for by hardware.

c) C programs make heavy use of procedure and function calls so that reasonable efficiency can be obtained only by using a call sequence with low overhead.

Aside from user-mode software, of course, much of the operating system itself would clearly have to be re-written for any new machine in order to accommodate its peripherals and process management hardware.

The work of transferring UNIX software to a Hewlett-Packard 21MX computer[3] would be minimized if the 21MX could be micro-programmed to emulate the PDP11. But the 21MX was not designed as a base for emulating other machines and it does a very inefficient job of emulating a PDP11. An alternative, which we describe in this memorandum, is to define a new machine whose architecture is similar to that of the PDP11 but whose instruction formats can be handled with minimum overhead by 21MX microprogram.

The need to define a new 16-bit machine code to handle C object programs provides an opportunity to improve upon the PDP11 machine code for this purpose. We have chosen to design a new machine, called the 'C machine,' so as to minimize the size of compiled C programs. To that end we obtained some statistics from programs compiled for the PDP11.

Statistics of C Programs

The PDP11 provides eight addressing modes.

R	Register R
(R)	Memory pointed to by R
(R++)	As (R) but increment R after addressing
((R++))	Memory pointed to by (R++)
(--R)	As (R) but decrement R before addressing
((--R))	Memory pointed to by (--R)
X(R)	Memory address X indexed by R
(X(R))	Memory pointed to by X(R)

The machine provides eight registers which are used by C programs as follows.

R0,R1	Temporary results
R2-R4	Local variables
R5	Stack frame pointer, FP
R6	Stack pointer, SP
R7	Program counter, PC

Some address forms have special uses. They are as follows.

X(FP)	Local variable or function argument
(PC++)	Literal constant
((PC++))	Literal pointer
(--SP)	Push item onto the stack
(SP++)	Pop item from the stack
X(PC)	Static variable

Table 1 contains some statistics obtained by scanning 21516 words of compiled C programs. In these programs, all branches, data-generating, pseudo-operation, and other non-data instructions were cast out. The special cases referred to above occur with the following frequencies.

X(FP)	18	per 100 instructions
(PC++)	31	per 100 instructions
((PC++))	1.8	per 100 instructions
(--SP)	9.4	per 100 instructions
(SP++)	7.6	per 100 instructions
X(PC)	15	per 100 instructions

If one excludes these cases one finds that most other address forms fit into one of the three patterns: R, (R), and X(R). Those that do not fit these patterns occur with the following frequencies.

(R++)	1.4	per 100 instructions
((R++))	0.04	per 100 instructions
(--R)	0.17	per 100 instructions
((--R))	0	per 100 instructions
(X(R))	2.0	per 100 instructions

The 21516 compiled words of program contained 11869 instructions and 9647 words of literals. Table 2 shows how the values of these literals were distributed. It will be seen that 4110 literals were in the range −7 to +7 so that some economy could be obtained by allowing small integer literals to appear in the instruction proper rather than in separately compiled words.

The C Machine

The 16-bit C machine is based upon the above statistics. Its central processor has 8 registers that are used as follows.

R0-R4 General-purpose registers; R0,R1 are used for all double-length and floating point operations.

R5 Stack frame pointer, FP; set equal to the stack pointer after a procedure call.

R6 Stack pointer, SP; points to the 'top' of the stack; as in the PDP11 a stack push decreases SP, and SP is used to save registers and control information during a subroutine call.

R7 This register reads as 0 and cannot be written into. In hardware it is the program counter, but it is not directly addressible.

A general address can take four different forms. It is a 4-bit field subdivided into a 1-bit indirection flag, I, and a 3-bit register address, R. The addressing modes are as follows.

I=1, R<7	register R
I=1, R=7	literal constant X
I=0, R<7	memory X(R)
I=0, R=7	memory X

Depending upon the instruction format the value X can either be in a 4-bit field of the instruction or can be in a word following the instruction.

The 21MX does not have a byte organized memory so that byte addressing in the PDP11 way would be quite expensive (in execution time) to implement. Byte addresses in the C machine have a word address in their least significant 15 bits and the most significant bit indicates which of two bytes in the word is intended. Thus the C machine address is equal to the PDP11 address after rotating the PDP11 address 1 bit to the right. To make this acceptable the C machine has separate instructions for doing pointer arithmetic. Although there are a few programs which make use of the particular structure of the PDP11 address word, this change in representation is not expected to affect program compatibility to any noticable degree. It also leads to more efficient programs when integer arrays are used.

There are five instruction formats. They are listed in detail in appendix A together with the operation codes available with each format.

a) Double operand instructions, with four fields:

A 4-bit function
A 4-bit general destination address
A 4-bit general source address
A 4-bit literal used to augment the source address

If the literal equals 15 then the value X used to compute the source address is obtained from (PC++) otherwise the value is the literal X−7.

b) Register/operand instructions, with four fields:

A 5-bit function
A 3-bit destination register address
A 4-bit general source address
A 4-bit literal used to augment the source address

c) Single operand instructions, with three fields:

An 8-bit function
A 4-bit general address
A 4-bit literal used to augment the general address

d) Jump instructions, with three fields:

A 7-bit function
A 2-bit type code
A 7-bit jump address

Depending upon the type code, this is either a 7-bit literal J, or is a 3-bit register R, and a 4-bit literal XX.

The type code can take three values:

1 jump to (PC)+J

2 jump to X(R), where X is (PC++) if XX=15 otherwise X is XX−7.

3 jump to (PC)−J

e) Miscellaneous instructions, which have only a function code in the least significant eight bits of the 16-bit word.

Implementation on the 21MX

Key samples of the microcode required to implement the C machine on the 21MX have been written; they are given in Appendix B. The general strategy is to use the switch jump micro-instructions to decode instruction fields. This leads to a large micro-program but a fast one. (It is not expected that the cost of micro-store will be significant.) It also means that we rely upon Hewlett-Packard's offer to use plug-in sockets for their jump table read-only memories. We would then replace the HP versions with our own. The main instruction interpretation loop has been written together with sample entries for the various switch jump tables. No code for interrupts is included but it is thought that that would be quite straight forward and not markedly different from the HP version. The same applies to the code required to handle the front panel.

The strategy used for Input/Output operations is to provide one C machine instruction whose operand is a legitimate HP I/O instruction. The micro-program simply transfers that operand to the instruction register and uses the HP micro-program sequence.

Memory addressing and protection have been cobbled into the 21MX in various places (presumably in an attempt to speed up the 2100 machine emulation). In particular there is hardware to handle the following.

a) addresses 0 and 1
These are used by HP to refer to the central processor registers A and B. If the micro-instructions that use the BAF and AAF flags are avoided, addresses 0 and 1 can be used normally. But they cannot be write-protected.

b) base page addressing
There are two 'fences:' one in the protection unit and one in the memory mapping unit. If they are both set to zero and mapping 'above the fence' is selected all addresses can be mapped.

c) illegal instructions
The memory address protection unit also checks for certain illegal instructions. Hewlett-Packard provide a strapping option on the protection board that can turn this function off. When that is done the micro-code must check for illegal use of I/O and HALT instructions.

It is apparent that Hewlett-Packard intend to encourage micro-programming by their 21MX users. Apart from the documentation, which is reasonably complete, they provide a writable micro-program memory module that can be loaded through the 21MX peripheral bus. Thus the most practical way of bringing up the micro-program for the C machine is to put two 21MX side-by-side. One 21MX would load the micro-program for the other. Eventually a HP provided read-only memory 'zapper' can be used to write the proper micro-program memory.

To further assist in the process of developing the micro-code HP have indicated that we could obtain one of their 'suitcase test units' which are really plug-in front panels for the micro-machine. There is a connector on the back of the CPU that allows one to operate the micro-machine directly. For example, from that connector one can load micro-code to be obeyed and one can alter the speed of the clock.

Conclusion

We have successfully designed a machine that can be used as a vehicle for transferring UNIX software to the Hewlett-Packard 21MX hardware. The software effort required to effect the transfer of C programs is quite minimal and that required for PDP11 assembly code would probably not be a major undertaking. Our experience in writing code for the 21MX indicates that that is not a difficult problem although it does require more than ordinary care. (Other related experiences suggest that prior thought pays unusually handsome dividends, by reducing debugging time, for this type of work.)

For word operations, the speed of the C machine would be somewhat less than a PDP11/40 but its speed for byte operations would be less than half that of the PDP11/40. Compiled C programs would be about 20% shorter on the C machine. A comparison of some C machine and PDP11/40 times appears in Table 3.

During the course of this study it has become evident how sensitive the performance of an emulated machine is to the way in which the micro-machine handles sub-fields of the instruction register. It is apparent that a versatile micro-machine must be able to extract and suitably shift arbitrary fields of an instruction. It must then be possible to deliver those fields to a central machine register and also to do a switch jump on the extracted value. These are expensive requirements and may mean that a general-purpose micro-machine is also an expensive machine. However, the comparative success of our study indicates that standardization at a language level only slightly above machine code may well leave one with sufficient latitude to plan on alternative hardware support for even quite low level software.

A. G. Fraser

MH-1273-$\frac{AGF}{DMR}$-UNIX
Att.
References
Tables 1-3
Appendices A, B

D. M. Ritchie

References

[1] Ritchie, D. M. and Thompson, K. The UNIX Time-sharing System. C. ACM 17, 7, (July, 1974), pp 365-375.

[2] Ritchie, D.M. C Reference Manual. TM-74-1273-1.

[3] "Microprogramming 21MX Computers: Operating and Reference Manual." Hewlett-Packard (#02108-90008), 1974.

TABLE 1
Frequencies of Addressing Modes for C Programs

The following table gives the numbers of instructions, in a sample of 11869 instructions, that used the indicated addressing mode and indicated register. The sample consisted exclusively of data-manipulating operations (moves, adds, and so forth); branches, calls, and the like were excluded.

mode	R0	R1	R2	R3	R4	R5	SP	PC
R	3315	509	587	878	1414	54	54	0
(R)	88	12	76	135	271	0	1739	0
(R++)	0	30	35	52	56	0	907	3794
((R++))	0	0	0	0	0	0	5	219
(--R)	0	0	6	5	10	0	1114	0
((--R))	0	0	0	0	0	0	0	0
X(R)	355	52	160	155	621	2224	0	1827
(X(R))	3	0	3	0	12	141	1	80

TABLE 2
Distribution of Literal Values in C programs

In the following table *P(n)* is the number of occurrences of the literal value *n* in the object program. *M(n)* is the number of occurrences of the value $-n$. *S(n)* is the number of occurrences of literals in the range $-n$ to *n* inclusive. *F(n)* is the fraction of the total number of literals that are in that range. In compiling this table, the literals in index words X(R) were divided by 2 to compensate for word rather than byte addressing.

n	*P(n)*	*M(n)*	*S(n)*	*F(n)*
0	86		86	1.3%
1	555	101	742	11.3%
2	1025	40	1807	27.5%
3	572	20	2399	36.4%
4	266	417	3082	46.8%
5	142	272	3496	53.1%
6	198	187	3881	59.0%
7	122	107	4110	62.4%
8	162	81	4353	66.1%
9	72	37	4462	67.8%
10	141	41	4644	70.6%
11	16	23	4683	71.1%
12	52	25	4760	72.3%
13	57	23	4840	73.5%
14	27	20	4887	74.2%
15	16	7	4910	74.6%
>15	1108	564	6582	100.0%

TABLE 3
Comparison of PDP11/40 and C Machine Times

In the following table *X* denotes a 16-bit literal while *x* denotes a literal in the range −7 to +7. A '$' prefix indicates that the operand is a literal constant rather than the content of a memory location. 'Mov' is a full-word data move; 'movb' is a one-byte transfer. The pairs of times for the byte moves are for the right- and left-byte cases respectively.

operation	operation times (microseconds)	
	11/40	21MX
mov R to R	0.90	1.95
mov $x to R	2.45	2.60
mov $X to R	2.45	3.25
mov x(R) to R	3.22	3.90
mov X(R) to R	3.22	4.55
mov R to X(R)	3.29	3.90
mov x(R) to X(R)	5.39	5.53
mov X(R) to X(R)	5.39	6.18
movb R to R	1.80, 2.14	3.58, 4.23
movb x(R) to R	3.56, 3.90	5.53, 6.18
movb X(R) to R	3.56, 3.90	6.18, 6.83
movb R to x(R)	3.43, 3.77	7.15, 7.48
movb R to X(R)	3.43, 3.77	7.80, 8.13
add R to R	1.14	1.95
add x(R) to R	3.51	3.90
add X(R) to R	3.51	4.55
add R to X(R)	4.10	4.88
add x(R) to X(R)	5.86	6.50
add X(R) to X(R)	5.86	7.15
function call	22.38	6.83
function return	18.76	6.50

APPENDIX A
Description of C Machine Language

Conventions

Eight of the HP registers are available to the user and are referred to herein as R0 through R7. They are used as follows.

R7 The program counter PC. R7 reads as zero when used in an instruction address field. R7 cannot be used as a destination.

R6 The stack pointer SP. SP points to the last entry put on the stack. It decreases as items are pushed on the stack.

R5 The stack frame pointer FP. It is set equal to the stack pointer immediately after a procedure call.

R4-R2 General purpose registers.

R01 General-purpose register used implicitly in double-length instructions is combination of R0 and R1.

Parentheses denote indexing or indirection. Thus Y(R) is Y indexed by R.

'++' denotes increment and is used to describe a side-effect after computing an address. Thus (PC++) refers to the word pointed to by the program counter and the program counter is then incremented.

'--' denotes decrement and takes place before the address is computed. Thus (--SP) is the word pointed to by the stack pointer after the stack pointer has been decremented.

The field structure of addresses and instructions is described by giving the name of each field followed by its size in brackets, for example R[3]. The field 0[9] is nine zeros.

Architecture

Address mapping and protection

The memory mapping unit can be made to give two full spaces, one for the user and the other for the system. The only special case is that locations 0 and 1 cannot be write protected.

Byte addressing

Two 8-bit bytes are stored in one 16-bit word. The most significant byte is in the least significant half of the word. A pointer is the address of a byte in main memory. It has the word address in its rightmost 15 bits; the top bit is set to address the leftmost byte in a word. Word operations ignore the leftmost bit of an address. When a byte is transferred to a register it occupies the right half of the register and the left half is cleared. It is not possible to address the leftmost byte of a register directly.

Input/Output

The HP Input/Output operations are preserved. The single C machine I/O instruction has an HP instruction as its operand. Since HP offers no byte-organized device that uses DMA there is no problem arising out of the different ways in which the HP 2100 machine and the C machine locate bytes in a word.

Addresses

address format (a)

field R[3] is a register address

address format (b)

fields: I[1],R[3]

	r-value	*l-value*
I=1	(R)	R
I=0	((PC++)+(R))	(PC++)+(R)

address format (c)

fields: I[1],R[3],X[4]
let Y = (if X = 15 then (PC++) else X−7)

	r-value	*l-value*
I=1 & R<7	(R)	R
I=1 & R=7	Y	null
I=0 & R<7	(Y+(R))	Y+(R)
I=0 & R=7	(Y)	Y

address format (d)

fields: I[2],R[3],X[4] or I[2],J[7]
let Y = (if X = 15 then (PC++) else X−7)

	jump addr
I=1	(pc)+J
I=2	Y(R)
I=3	(pc)−J

Instructions

double operand instructions

fields: F[4],D[4],S[8]
D is format (b) address; S is format (c) address
$4 \leqslant F < 12$.

D = S, S = D	word assignment
D = D+S, D = D−S	integer arithmetic
D = D | S, D = D&S	logical operations
test (S&D), test (S−D)	word comparison

register-operand instructions

fields: F[5], D[3], S[8]
D is format (a), S is format (c), $24 \leqslant F < 32$

D = D+S, D = D−S	pointer arithmetic
test (S−D)	pointer comparison
D = S	load byte
S = D	store byte from ls of D
test (S−D)	byte comparison
shift D by S places	logical shift
rotate D by S places	

single operand instructions

fields: F[8], S[8]
S is format (c), $1 \leqslant F < 32$

(--SP)=S, S=(SP++), S=0	word transfer
(--SP)=S, S=(SP++), S=0	S is byte
(--SP)=S, S=(SP++)	word S in user space
(--SP)=S, S=(SP++)	byte S in user space
S = S+1, S = S−1, S = −S	integer arithmetic
S = S+1, S = S−1	pointer arithmetic
test S	compare word with zero
test S	compare byte with zero
rotate S by 8 bits	
R01 = R0 * S	multiply
R0 = R01 / S	with remainder in R1
R01 = S, S = R01	double word assignment
R01 = R01+S, R01 = R01−S,	floating point
R01 = R01*S, R01 = R01/S	floating point
test (R01−S)	floating point
shift R01 by S places	arithmetical shift
shift R01 by S places	logical shift
rotate R01 by S places	
execute I/O instruction S	

jump instructions

fields: F[7], J[9]
J is format (d), $16 \leqslant F < 32$.
Jump conditions are determined by previous arithmetic, test or logical operation.

jump on zero, on non-zero
jump on negative, on non-negative
jump if carry set, if not set
jump if overflow set, if not set
jump if less than zero (neg. or ovr. not both),
jump if not less than zero
jump if less than or equal to zero
jump if greater than zero
jump unconditionally
function call; saves PC, FP, R4, R3, R2
interrupt call J (I=1 only); saves all regs and flags
system call J (I=3 only)

miscellaneous instructions

fields: 0[9], F[7]; $0 \leqslant F < 128$

return from function
return from interrupt
return from system
set priority level
halt
reset peripheral devices

Bell Laboratories

subject: **Changes in the C Environment for UNIX/TS Edition 1.0**

date: **September 27, 1978**

from: **Andrew Koenig**
MH 8234
2C-258 x5570
MF78-8234-84

MEMORANDUM FOR FILE

0. INTRODUCTION

This document describes differences users may encounter when changing to UNIX/TS Edition 1.0 from the last release of the C compiler shipped for Generic 3. It is intended as a conversion aid, so the emphasis is on incompatibilities rather than new facilities.

Please note that this document is only a *guide*; refer to the *UNIX/TS User's Manual* for complete information.

1. LIBRARY CHANGES

The changes that are most likely to be noticed are in the run-time library. The "Standard I/O Library" has been incorporated into */lib/libc.a* along with the contents of */lib/liba.a*; this latter library is gone. *Printf* has been rewritten into portable C; there are a few incompatibilities with the old version. Finally, there are a number of smaller changes and incompatibilities.

1.1 Environments

The system now makes available to the user program a table of *environment variables*. Each variable has a name and a value; both name and value are character strings. The values of environment variables are preserved across *fork* and *exec*; they can also be altered easily using the Shell and somewhat less easily using the new *execle* and *execve* system calls.

The new *getenv* function can be used to retrieve the value of an environment variable.

1.2 The Standard I/O Library

In the past, there were two I/O libraries available. One was documented by *A New Input-Output Package* (Ritchie), and was made available through the −lS loader option. The other, older one was made available whenever a C program was being compiled; it was characterized, among other things, by use of the names *fin* and *fout* to control disposition of standard input and output files.

The older library has now vanished, along with the −lS option. All programs will receive the new I/O library without any explicit action. In addition, the libraries obtained by −lc and −la have been merged; this combined library is accessed (where needed) by −lc or −l.

1.3 Printf

In the interests of portability, *printf* has been rewritten into portable C. This results in load modules some 1800 bytes larger than previous versions.

The correct way to write a long integer is now %ld or %lo; the previous forms D and O are going away. The purpose of this is to permit X, E, and G format codes for indicating that the letters

produced by the format code are to appear in upper case.

The %r format code has been removed; if you don't know what it was, you don't want to.

1.4 Scanf

The way in which *scanf* treats white space has changed slightly. No longer is it the case that *scanf* will skip white space in the input for each character in the format. Rather, a space, tab, or newline in the format will match optional white space in the input. Thus

```
"alpha = %d"
```

will match any of

```
alpha=12
alpha =12
alpha= 12
alpha = 12
```

but not

```
a lpha=12
```

as was formerly the case. Note that this change may require white space to be inserted in format strings of formerly working programs to maintain compatibility.

1.5 Mathematical Routines

The mathematical subroutines have been moved to a separate library obtainable by the **-lm** option. Declarations for these routines can be obtained in **<math.h>**.

1.6 Character class routines

The routines that test character class (*isdigit*, etc.) are no longer defined in **<stdio.h>**; rather, they are defined in **<ctype.h>**. Thus, a line of the form

```
#include <ctype.h>
```

will have to be added to those programs which use *isdigit*, *isupper*, *islower*, and their relatives.

1.7 Error Recovery

The library now incorporates two new routines, *ssignal* and *gsignal*. In the future, these routines will be used by other routines in the library to cause automatic program termination on detection of various common errors, with the possibility of finer control as a user option.

This description is deliberately vague, as the facility is still in the planning stage.

1.8 Time of Day

There is a new function *tzset*. It is called with no arguments, and looks for an environment variable TZ. This variable is expected to be in the form EST5 or EST5EDT and represents the difference between the local time zone and GMT, surrounded by the names of the local and (optional) daylight time zones. If *tzset* finds an environment variable TZ in this form, it sets the time zone parameters *timezone*, *tzname*, and *daylight* appropriately. *Tzset* is now called automatically by *asctime*, so it usually need not be called by the user.

1.9 Miscellaneous

1.9.1 chown. *Chown* now takes three arguments: the file name, the new owner, and the new group. This is necessary because owner and group can now each be up to 16 bits.

1.9.2 tell. *Tell* is gone; *lseek* instead returns a value indicating the location sought.

1.9.3 setexit and reset. *Setexit* and *reset* are gone; their function is taken over by *setjmp* and *longjmp*. These new routines provide all the facilities of *setexit* and *reset* in a more general form.

1.9.4 nargs. *Nargs* is gone. There is no replacement routine, as *nargs* cannot be made to work with separate I and D space.

1.9.5 String routines. *Strcatn*, *strcpyn*, *strcmpn*, *index*, and *rindex* have been renamed *strncat*, *strncpy*, *strncmp*, *strchr*, and *strrchr*, respectively. This follows the recommendataions of the C Standards Task Force, and also allows compatibility with systems that require distinct external names to differ within their first six characters.

1.9.6 Effective user and group id. There are two new routines, *geteuid* and *getegid*, which return the effective user and group ID, rather than the real user and group ID.

1.9.7 time. The *time* routine now returns a **long** value; it will also store a copy of the value in the (long) location addressed by its argument unless that argument is **(long *)0**.

1.9.8 The password file. The format of */etc/passwd* has changed slightly with the introduction of UNIX/TS; this change is reflected in the various routines which extract information from */etc/passwd*. In addition, a new file, */etc/group*, has been created to hold information about group access privileges. This file is searched by a new set of routines.

The names of the routines under discussion are:

endpwent	endgrent
getpwent	getgrent
getpwnam	
getpwuid	getgrgid
setpwent	setgrent

2. THE LANGUAGE

2.1 The Preprocessor

John Reiser has rewritten the C preprocessor. The new one is largely compatible with the old one, and much faster, but there are a few changes.

2.1.1 General. Symbols defined on the command line by -D*foo* are defined as **1**, i.e., as if they had been defined by

```
#define foo 1
```

or

```
-Dfoo=1
```

This means that names automatically defined by the preprocessor (specifically *unix* and *pdp11*) cannot be used as identifiers in the program without naming them in **#undef** statements or using the **-U** preprocessor option.

The directory search order for **#include** requests is

1. the directory of the file which contains the **#include** request (e.g. **#include** is relative to the file being scanned when the request is made), for statements of the form

   #include "*name*"

2. the directories specified by **-I**, in left-to-right order (as usual, the null string can be used to name the current directory)

3. the standard directory(s) (which for the UNIX system is */usr/include*)

An unescaped newline terminates a character constant or quoted string.

An escaped newline (a backslash immediately followed by a newline) may be used in the body of a **#define** statement to continue the definition onto the next line. The escaped newline is not included in the macro body.

Comments are uniformly removed (except if the argument **-C** is specified). They are also ignored, except that a comment terminates a token. Thus

```
foo/* la di da */bar
```

may expand 'foo' and 'bar' but will never expand 'foobar'. If neither 'foo' nor 'bar' is a macro then the output is the string 'foobar', even if the preprocessor name 'foobar' is defined as something else. The file

```
#define foo(a,b)b/**/a
foo(1,2)
```

produces '21' because the comment causes a break which enables the recognition of 'b' and 'a' as formals in the string "b/**/a".

Macro formal parameters are recognized in **#define** bodies even inside character constants and quoted strings. The output from

```
#define foo(a) '\\a'
foo(bar)
```

is the seven characters " '\\bar'". Macro names are not recognized inside character constants or quoted strings during the regular scan. Thus

```
#define foo bar
printf("foo");
```

does not expand 'foo' in the second line, because it is inside a quoted string which is not part of a **#define** macro definition.

Macros are not expanded while processing a **#define** or **#undef**. Thus

```
#define foo bletch
#define bar foo
#undef foo
bar
```

produces 'foo'. The token appearing immediately after an **#ifdef** or **#ifndef** is not expanded (of course!).

Macros are not expanded during the scan which determines the actual parameters to another macro call. Thus

```
#define foo(a,b)b a
#define bar hi
foo(bar,
#define bar bye
)
```

produces " bye" (and warns about the redefinition of 'bar').

2.1.2 Bugs fixed.

1. "1.e4" is recognized as a floating-point number, rather than as an opportunity to expand the possible macro name "e4".
2. Any kind and amount of white space (space, tab, linefeed, vertical tab, formfeed, carriage return) is allowed between a macro name and the left parenthesis which introduces its actual parameters.
3. The comma operator is legal in preprocessor **#if** statements.
4. Macros with parameters are legal in preprocessor **#if** statements.
5. Single-character character constants are legal in preprocessor **#if** statements.
6. Linefeeds are put out in the proper place when a multiline comment is not passed through to the output.

7. The following example expands to "# # #" :

```
#define foo #
foo foo foo
```

8. If the -R flag is not specified then the invocation of some recursive macros is trapped and the recursion forcibly terminated with an error message. The recursions that are trapped are the ones in which the nesting level is non-decreasing from some point on. In particular,

```
#define a a
a
```

will be detected. (Use "#undef a" if that is what you want.)

9. The recursion

```
#define a c b
#define b c a
#define c foo
a
```

will not be detected because the nesting level decreases after each expansion of "c".

10. The -R flag specifically allows recursive macros and recursion will be strictly obeyed (to the extent that space is available). Assuming that -R is specified:

```
#define a a
a
```

causes an infinite loop with very little output. The tail recursion

```
#define a <b
#define b >a
a
```

causes the string "<>" to be output infinitely many times. The non-tail recursion

```
#define a b>
#define b a<
a
```

complains "too much pushback", dumps the pushback, and continues (again, infinitely).

2.1.3 Stylistic choice.

1. Nothing (not even linefeeds) is output while a false #if, #ifdef, or #ifndef is in effect. Thus when all conditions become true a line of the form '# 12345 "foo.c"' is output.
2. Error and warning messages always appear on standard error (file descriptor 2).
3. Mismatch between the number of formals and actuals in a macro call produces only a warning, and not an error. Excess actuals are ignored; missing actuals are turned into null strings.

2.1.4 Incompatibility. The virgule '/' in "a=/*b" is interpreted as the first character of the pair "/*" which introduces a comment, rather than as the second character of the divide-and-replace operator "=/". This incompatibility reflects the recent change in the C language which made "a/=*b" the legal way to write such a statement if the meaning "a=a/ *b" is intended.

2.2 The Compiler

2.2.1 Enumerated Data Types. Enumerated data types are here, though not yet documented, so that **enum** is now a keyword.

2.2.2 Unsigned numbers. The value returned by *sizeof* is now *unsigned* rather than *int*, so care must be exercised in the use of **sizeof** in a few strange cases. For example, the following no longer works:

```
if (n < - sizeof (x)) { ... }
```

because unary — is meaningless when applied to an **unsigned** value.

2.2.3 Structure and Union Assignments. It is now possible to assign structures and pass them as arguments and results of procedures. This feature is not new in the latest release, but it is sufficiently important that it is worth noting anyway.

3. SOURCE STRUCTURE

The new preprocessor and changes in the library make the source structure of this new release of C different from previous versions.

3.1 The Compiler

The new preprocessor is comprised of three source modules: *cpp.c*, *cpy.y*, and *yylex.c*. *Cpy.y* should be processed by *yacc* to produce *cpy.c*; this and *cpp.c* should then be compiled together to produce the preprocessor. Despite its name, *yylex.c* does not involve using *lex*, and it is not directly compiled; rather, it is named by **#include**s in the other modules.

3.2 The Library

The source for the mathematical routines in the C library is now in */usr/src/lib/libm*. The source in */usr/src/lib/libc* is now organized in five subdirectories:

1. *crt*, which contains run-time routines that are invoked by generated object code without ever being explicitly referenced by the programmer. These routines are largely in assembler language, and do things like **long** multiplication and division.
2. *csu*, which contains routines that are explicitly referenced by the *cc* command; these routines are used for run-time initialization.
3. *gen*, which contains those routines described in section 3 of the manual that are not part of the "standard I/O package",
4. *stdio*, which contains those routines described in section 3 of the manual that *are* part of the "standard I/O package", and
5. *sys*, which contains the routines described in section 2 of the manual. These routines are all in assembler language, and are interfaces between the C language and the UNIX system calls.

The other files in the */usr/src/lib/libc* directory are used as part of the installation procedures. *Order.in* and *order.out* are used to define the ordering of the modules in */lib/libc.a*, and *libc.rc* is a command file to recompile the library.

MH-8234-ARK-troff

Andrew Koenig

Copy to
Members of Department 8234
Members of the C Standards Task Force
M. P. Fabisch
H. T. Gibson
N. A. Martellotto
B. A. Tague

Berkeley Software (Pascal and "ex") for UNIX+

A package of software for UNIX is available from the Computer Science Division of the University of California at Berkeley. This package includes the instructional Pascal system which has been in use at Berkeley for over two years and the editor "ex", an extension of the editor "ed". (Newer software is available also... see page 3)

Both "ex" and the Pascal system will run on 11/{34,40,45,60,70} CPU's. A two process version of the Pascal translator "pi" is used on 11/{34,40,60} which do not have the separate I/D space capability. To run these programs a full load of user core (64K bytes) is required.

Source code, binaries and machine readable versions of all documentation are included with the tape. The Pascal system and the "ex" text editor are distributed under the attached agreement; UC Berkeley is thus the sole source for this software.

The distribution tape is a standard "tp" format, 800BPI (1600BPI available on request) magnetic tape. We will supply a 1200 foot magnetic tape on which the software will be written. RK cartridge distribution of the software is NOT available.

To receive the tape fill out the attached agreement and send it with

1. A CHECK (purchase orders are NOT acceptable) for $60 US
2. A copy of your UNIX license agreement
3. A clear indication of your mailing address
4. A filled in copy of the agreement (which follows)

to

Berkeley UNIX Software Distribution
c/o William N. Joy
Computer Science Division
Department of EECS
Evans Hall
University of California, Berkeley
Berkeley, California 94720

Included with the tape are copies of all documentation for the programs on the tape. If you wish additional copies of the documentation, send $5 for each additional copy. (You can also get just the documentation, in this case you need only send $5 and a copy of your UNIX license).

If you have received an earlier version of this tape (dated January 16, 1978), you can get the latest (January 79) version of this tape by sending a 1200' tape to me. There are some bug fixes in the Pascal system on this new tape and the new tape includes a Pascal for non-separate I/D machines which the Jan. 78 tape did not.

Questions about this tape can be directed to William Joy at the address above or at (415) 642-4948. Messages can be left at (415) 642-1024.

+ UNIX is a trademark of Bell Laboratories.

Software Agreement

Name of Receiving Institution:

1. We have received from the University of California, Computer Science Division, Department of Electrical Engineering and Computer Science, the (first) Berkeley Software Tape containing the Pascal system and the "ex" editor (version 1.1) and other software, for a one time handling charge of $60.

2. We declare that we will use this software for in-house applications, research and development only.

3. We commit ourselves not to pass the specified software and documentation or parts of it to other parties outside our establishment.

4. We accept that the University of California has no committments concerning software service or debugging.

5. We commit ourselves to inform the University of California about faults detected in the software.

6. We commit ourselves to inform the University of California about improvements we make to the received software.

7. We commit ourselves to name the University of California as a reference in publications and descriptions if the received software or documentation is an essential part of a basic idea of the work described in the paper.

I accept the above conditions.

Signature: Date:

Position:

New Berkeley Software

Some new software developed at UC Berkeley will be available after March 1, 1979. The major new software included here will be:

-- A new version of the "ex" editor

This new version improves the terminal driving capabilities of the editor to provide screen editing facilities using intelligent terminals. All terminal capabilities are describe in a file so that the editor can drive new terminals after their descriptions are edited into the file. Padding and other requirements of intelligent terminals are dealt with in the editor. The editor is particularly designed so as to be usable as a screen editor on low speed dialup lines with intelligent and unintelligent terminals. The size of the editing window can be made small, and will expand as you work. This prevents long delays, e.g. after a search when the window will be redrawn in its smaller size.

The screen editing facilities of the editor have been extended so that it is possible to perform editing completely within "visual" or "open" modes. Open mode has been extended so that it works on hardcopy terminals.

Other new facilities include:

-- A facility for continuous text input with the editor breaking over new lines automatically at spaces near the right margin.

-- The capability to filter parts of the editor buffer through specified commands, and to read/write parts of the buffer from/to command.

-- Text registers (a-z) into which lines may be placed and carried over when editing new files.

-- A "tags" facility for editing large programs which are broken into many files. The editor can be told the name of a tag (usually a function name) and will then switch to the file where that function resides and position itself at that function.

While "ex" edits only one file at a time, it remembers the line position in the previous file, so that it is possible to easily switch back and forth between two files.

The new version of ex requires separated I/D space and will no run on 11/{34,40,60} CPU's.

-- A new shell "csh"

This shell incorporates good features of the earlier shells, previous Berkeley shells and the version 7 shell. It also has a history mechanism (similar to that of INTERLISP) so that previous commands can be repeated and/or corrected. This shell runs on both version 6 and 7 systems. The shell does not require separate I/D space.

-- A new macro package for nroff/troff "-me"
This macro package is easy to use and especially easy to adapt to different formats. It works both with nroff and troff and provides a large number of hooks for special requirements. A beginners introduction and a reference manual are also available.

-- A new "Mail" program
This new mail program uses "mail" to do the actual mailing and concentrates on providing a simple and friendly environment for processing large amounts of mail.

-- Modifications to the stdio package to allow simultaneous read/write

If you have obtained a copy of the Berkeley Software tape then all you need to do is send a 1200' tape to the address below and we will send you this software after March 1, 1979. In order to get the new nroff/troff macro package and the "stdio" modifications, you must include a copy of your license for the version 7 phototypesetter package.

If you have not obtained the Berkeley Software tape then you should send $25 and a 1200' magnetic tape along with a copy of your license and a filled in copy of the agreement (on page 2) to the address given on page 1.
%

University of Hawaii at Manoa

Institute for Astronomy
2680 Woodlawn Drive • Honolulu, Hawaii 96822
Telex: 723-8459 • UHAST HR

February 5, 1979

Mr. Ian Johnstone
Australian Graduate School of Management
University of New South Wales
Kensington 2033, Australia

Dear Mr. Johnstone:

At the recent West Coast UNIX Meeting, I had a chance to talk with John Lions about how I might best set up my new UNIX installation (PDP 11/40, 15 MB RK05-type disk storage, one TJU16 mag tape). In view of the difficulties that the U.S. users group is facing (problems of size, tax laws, incorporation, lawyers, etc.,) he suggested I write to you and see if it is possible to obtain a copy of what he called your V6.5 distribution tape. I am particularly interested in a version of V6 UNIX that has the 101 fixes (which I have only heard about - I have no diff listing), a C compiler that produces code that uses the 11/40 floating point hardware, and a linker that can handle fortran overlays.

A proof of V6 license is enclosed. I will be glad to send you dollars or a blank tape or both. If you can help, I will be most grateful.

Sincerely,

Richard J. Wolff

Richard J. Wolff
Associate Astronomer

RJW/mm
Enclosure

UNIVERSITY OF KENT AT CANTERBURY
COMPUTING LABORATORY

DIRECTOR:
E. B. SPRATT. B.Sc., Ph.D., M.B.C.S., F.I.M.A.
PROFESSOR OF COMPUTER SCIENCE:
P. J. BROWN. B.A., M.A., Ph.D., F.B.C.S.
TELEPHONE 66822 EXT.7619

THE UNIVERSITY
CANTERBURY
KENT
CT2 7NF

RPAC/PJB

9th February, 1979.

Mr. Ian Johnstone,
A.G.S.M.
University of New South Wales,
P.O.Box 1, Kensington,
N.S.W.
AUSTRALIA 2033.

Dear Ian,

I was idly looking through part of your massive software distribution of last year when I came across a directory called 'v6unix' which contains a new 'Bell' labs UNIX base system. Unfortunately, part of the 'diff' file pertaining to 'sysl.c' is corrupt. I would be very grateful if you could send me a listing of your source. There appears to be a block or so missing about 17 lines after the line:

```
1ØØ,1Ø6c144,152
```

in the section headed

```
------ ken/sysl.c
```

If you get a lot of garbage out then your source file is corrupt too.

I would like to know whether you have looked at these changes and rejected them for any reason. There appears to be a version of LOWER_TEXT_SWAPS which allows text and data to be loaded into non-contiguous areas, which seems to be a good idea. There is also a new scheduler which swaps in depending on 'p nice' as well as 'p time'.

Another interesting thing is that the condition in setpri() - (line 2165 in J.Lions standard UNIX work) is reversed, reading

```
if(p < curpri)
```

J.Lions is a bit non-commital on this point - leaving it to be "worked out by the reader" - what do you think?

All this has led me to wonder about scheduling priorities. But it is very difficult to actually work out whether things are getting "better" without some form of bench mark. Have you got access to one?

I would like to be able to stop the unfair situation of a person running a large program hitting all the other users of the machine. I feel that the user should be affected and not the poor people trying to edit files. I think that one solution to this may be to add 1 to p_nice on every fork (unless the parent pid is 1 ??). But it is very difficult to find out if this really would produce a "better" system. Has anybody in Australia mucked about with process scheduling and produced any definative results?

continued.....

This letter is all questions and no answers. Thanks for the distribution. I haven't got all of it yet as we haven't got mag tape and getting lots of RKØ5's from Scotland to Kent is not easy!

I hear on the ubiquitous grapevine that John Lions is seconded to Bell for a year. Has he got any news of UNIX 7 and or VAX UNIX, there seems to be so many rumours flying about here a word from the "horse's mouth" would not go amiss.

Yours sincerely,

Peter Collinson

R. P. A. Collinson
Lecturer in Computer Science

TELEPHONE 370 0111
TELEX—UNIVQLD AA40315
TELEGRAMS—BRISBANE UNIVERSITY

YOUR REFERENCE:

OUR REFERENCE:

University of Queensland

ELECTRICAL ENGINEERING
DEPARTMENT

ST. LUCIA, BRISBANE
AUSTRALIA, 4067

13th February, 1979.

Mr Ian Johnstone,
Australian Graduate School of Management,
University of N.S.W.,
KENSINGTON, N.S.W. 2033.

Dear Sir,

Further to our telephone conversation this morning.

I am sending a magtape and would appreciate it if you can provide us with the following:-

(i) Your local Plot/Graph routines and libraries

(ii) The Plot 10 Graphics package. We have a licence here at Qld. Uni for the Tektronix TCS software.

(iii) Can you supply further details on the Known Fortran IV bugs.

Yours faithfully,

C. Harridge

(C. Harridge),
Electrical Engineering.

UNIVERSITY OF GLASGOW

TEL: 041-339 8855
EXT. 478/7458

Computing Science Department.
THE UNIVERSITY,
GLASGOW, W.2

14th February, 1979.

Dr. Ian Johnstone,
Australian Graduate School of Management,
University of New South Wales,
P.O. Box No. 1,
Kensington,
New South Wales,
AUSTRALIA 2033.

Dear Ian,

I have enjoyed re-reading my previous letters (plus much else of interest!) in your latest newsletter. I am enclosing herewith a tape containing the latest release of the Sussex POP-2 System (dubbed "POP-11"). I have not yet received the hoped-for replacements for the corrupted QMC files. When these arrive I will send them, plus the other software referred to in my last letter, on your own tape. I would be glad if you could return my tape intact when you have copied it.

Best wishes.

Yours sincerely,

Alistair C. Kilgour

Alistair C. Kilgour.

encl.

PRENTICE COMPUTER CENTRE
UNIVERSITY OF QUEENSLAND
ST. LUCIA, QUEENSLAND, AUSTRALIA, 4067
TELEPHONE: 3773021
TELEX—UNIVQLD AA40315

DIRECTOR:
ALAN W. COULTER
B.ECON., F.A.C.S., M.A.PS.S.

AKH:GJ

13 February, 1979.

Mr. Ian Johnstone,
A.G.S.M.
P.O. Box 1,
KENSINGTON.
N.S.W. 2033

Dear Ian,

The University of Queensland runs a DECsystem1090 and a large number of PDP11s. We wish to provide the PDP11 users easy access to the facilities of the DEC10. This can be done relatively easily provided the PDP11 has an implementation of either DECNET or NCL (the DEC10 network protocols).

I am writing to you to ask if you or any of the readers of your newsletter have or know of a UNIX implementation of DECNET or NCL.

Yours faithfully,

Arthur Hartwig
Systems Programmer

CSIRO

CENTRAL INFORMATION, LIBRARY AND EDITORIAL SECTION

314 ALBERT STREET, *P.O. BOX 89*, EAST MELBOURNE, VIC. 3002 TELEPHONE 419 1333 TELEGRAMS *CORESEARCH MELBOURNE*

JS/SS

14th February, 1979.

Mr. Ian Johnstone,
A.G.S.M.,
P.O. Box 1,
KENSINGTON, N.S.W. 2033.

Dear Ian,

Thank you for the October and December issues of the Australian UNIX Users' Group Newsletter. I would like to receive it on a regular basis.

Yours sincerely,

John Shortridge

John Shortridge.

UNIVERSITY OF GLASGOW

TEL: 041-339 8855
EXT. 478/7458

Computing Science Department,
THE UNIVERSITY,
GLASGOW, G12 8QQ.

12th March, 1979.

Dr. I. Johnstone,
Australian Graduate School of Management,
P.O. Box 1,
Kensington,
New South Wales,
AUSTRALIA 2033.

Dear Ian,

Thanks for your letter and the notes from John Lions. I shall send a copy to a few people in the U.K. who I know may be interested - I hope this is O.K. even although it is an "unofficial" report. Bruce Anderson at Essex is now editor of the U.K. Newsletter. I think he is waiting till after the U.K. User Group Meeting at Canterbury on 29th & 30th March (to be addressed by Ken Thomson and Brian Kernighan) before putting together the next issue.

Your tape arrived safely, but our tape drive is out of action at the moment. We have recently traded in our 11/40 for a second-hand 11/45, and it seems that the total number of perpherals it can support at any time is always N-1, where N is the number supported by the 11/40. At the moment the - 1 is the magnetic tape and because of lack of space on the bus it may be quite a while before it gets fixed. I shall try to get your tape processed somewhere else so that I can distribute the contents at the Canterbury meeting.

The comments on the last page of John Lion's report about possible difficulties with instruction recovery on the 11/60 are a bit worrying. I know of several people who have ordered 11/60's with a view to running UNIX. Perhaps they are in for some disappointment?

Last week we visited the Computer Science Dept., at Edinburgh to see how they were getting on with their VAX machine. They have a megabyte of store (which you can just about see without a magnifying glass) but only one DZ11 with 8 terminals. They are most unimpressed with VMS, in particular the filing system, which appears to be extremeley inefficient. Jeff Tansley at Edinburgh has just had word from Western that Version 7 licences are now being issued for VAX machines - don't get trampled in the rush. They are getting UNIX for evaluation purposes, but because of a strong case of the "not invented here" syndrome. I feel that in the long run they will find reasons for rejecting it and come up with something of their own. (Which may be no bad thing).

Backing/

Backing store on the Edinburgh machine consists at present of two RK07 drives, but they have ordered two 300 Mbyte CDC drives with a controller supplied by Systems Industries. Initially this will go on the Unibus, but SI are designing an SBI interface which they hope to have ready in under a year from now.

Our new System Manager Zdrav Podolski is going to handle software distribution, etc., from now on, so perhaps you could send tapes or requests for software direct to him.

Best wishes.

Yours sincerely,

Alistair C. Kilgour

Alistair C. Kilgour.

THE UNIVERSITY OF NEWCASTLE
NEW SOUTH WALES, 2308

DEPARTMENT OF MATHEMATICS

TELEPHONE 68 0401
EXT. 596

BC:JL

27th March, 1979.

Dr. I. Johnstone,
Australian Graduate School of Management,
University of New South Wales,
P.O. Box 1,
KENSINGTON, N.S.W. 2033.

Dear Ian,

As you know, we are now running UNIX on the PDP 11/45 in the Mathematics Department at Newcastle. This machine is used both for research and teaching, it currently consists of

PDP 11/45 C.P.U. (with Floating Point)
64K Dec Memory
RS04 Fixed Head Disk
3×RK05 Disk Drives
2×TC11 Dectape Drives
CMS11 Card Reader
LP11 Line Printer
LA36 Terminal
VT55 Terminal
VC404 Terminal (Anderson Digital Electronics)
2×LA30 Terminal

Orders have also been placed for 64K DATARAM add on memory and eight more terminals.

I have included a magtape on which I hope you can send us the software detailed below.

D211 device driver
CMS11 card reader driver
DEC Fortran IV (RT11 fortran)
BASIC-PLUS

Photocopies of the latest invoices and orders for BASIC-PLUS and FORTRAN IV are included to show that we have software agreements for them.

Could you please write the tape in "TP" format, this makes life much easier for me.

Thanks for your time and trouble.

Yours sincerely,

B Check

TELEPHONE:
or: 692 2455

The University of Sydney
THE DEPARTMENT OF CHEMICAL ENGINEERING
N.S.W. 2006

March 28th, 1979

Dr. I. Johnstone,
A.G.S.M.,
The University of New South Wales,
P.O. Box 1,
KENSINGTON, N.S.W. 2033

Dear Ian,

In response to your request for Unix site configuration, I produced the following note.

Site:	Dept. Chemical Engineering, University of Sydney 2006
CPU:	PDP-11/20 + 56 K bytes core memory.
Disks:	1 Pertec DM3000 Disk (logically 4 x RKØ5 drives)
Terminals:	4 teletypes (2 x 33's, 2 x 43's)
Other Peripherals:	Serial link (homemade to the Faculty of Engineering 11/45 (RSX-11D))
	DU-11 to Cyber (not operational as we do not have the core to run as a 200 user terminal).
	PC-11 reader/punch
Software:	MX on Mini-Unix

I have included a few notes on MX that you may find assuming.

Future development of Mini Unix include its transportation to PDP 11/03's using flappy disks as system hosts.

This work is about 80% completed, and is looking very promising. Who knows, the 11/03 may yet prove to be almost as fast as an 11/20.

Cheers,

CD Rowles

Chris Rowles.

MINI UNIX - WHAT IS IT?

Mini unix is a unix system implemented on an unmapped processor. It consists of a software package to emulate the instruction set of the PDP-11/45, upon which the kernal elements of the system resides.

These then form the major implementation differences between Unix and Miniunix, and unfortunately, the difference shows. The emulator makes the system very slow. Preliminary tests indicate that there is a speed difference factor of about 10 to 15 between Miniunix running in an 11/45 and on 11/20. Heavy Fortran floating point calculations increase this difference to between 35 to 60 depending upon the calculations involved. These tests were carried out in single user mode on RK05 disk drives. The main non-mathematical oriented test is recompiling a new system configuration (8 to 10mins on 11/45 and 60 to 125 mins. on an 11/20).

The unmapped nature of the system leads to some very serious side effects. The most notabel being the lack of protection against malfunctioning user programs. This leads to unexpected system crashes. It also give rise to some unexpected behaviour when high priority processes compete for system resources.

In short Mini unix appears as an amazing product - functionally exceedingly rich and flexible; operatively a real competitor with NOS 2.1 running Telex on a Cyber 72-26A, as a fast timeshared system.

Functionally it bears a very strong family resemblance to UNIX. In detailed implementation, it is very different. The main differences are summarised below:-

.a. Only a single user program may be core resident at one time.

.b. Pipes are implemented as disk files, communicating between sequentially running programs, rather than programs that compete for system resources.

e.g. nm /mx pr -w132 -5 -h "system symbol table" /dev.lp

is implemented as:

1. nm producing a 'pipe' file on the disk

   then

2. pr printing & formating the output 'pipe' from nm and sending its output to the line printer.

In both Unix, both nm & pr would be sharing memory as a result of the single piped command to the shell.

.c. User programs are limited in size to (64K-TOPSYS) bytes, where TOPSYS, is the size of the system kernal space.

.d. User program (executable images) are not transferable between MINI-UNIX systems of different kernal sizes. This means that programs running on a 12K system must be recompiled to run under a 14K system.

.e. Memory management separation of system & user space is not available. Hence it is possible for a user to wipe the entire operating system.

.f. No optimisation is applied to disk drives.

.g. Swapping algorithm is very simple minded, indeed. Hence the system is very dependent upon the priority level at which a task is executed.

System Re-configuration:

As delivered Mini-unix supports one terminal & two RK.05 disk drives. Ths user must reconfigure it for any other devices he desires to support. This is a 'simple' editing procedure, involving two files:-

low.s	-	low core map.
conf.c	-	system driver entry table.

The new 12K system can then be developed by linking the new executive.

This simple procedure has several hidden traps, for new players and established UNIX GURU's. It is not as straight forward as it is made out to be.

The undocumented assumptions include:-

a) System bootstrap is located at 173ØlØ. (not true for some older Dec. diode bootstrap roms, which have a habit of starting at 1731ØØ).

A system crash causes a core dump to rkØ and an automatic reboot by jumping to 173Ø1Ø. If you do not have a rom at 173Ø1Ø, this causes a second memory trap to an unspecified section of code (the low core Ø2 through 2Ø being used to save the contents of the processor registers).

b) The root device is rkØ and will never be changed.

c) The swap device is also rkØ and will not be changed.

Both of these conditions appear to be user selectable by altering ROOTDEV SWAPDEV unless the system source is edited to correct this oversight.

d) The editor does not understand about links. Specifically all of the .h files uses by the system occur in two separate directories, /usr/sys & /usr/sys/mxsys. Half of the system use one directory and the other half use the other directory. Changing files in one directory does not alter the 'identical' file in the other file.

e) The system needs at least 4 buffers, other than those assigned to the superblocks of the root device and mounted file system, in order to operate. If less than 4 buffers are available, the system stands a good chance of locking up with every process sleeping waiting for a free buffer.

f) The system panic routine must be altered to update the disk before it dumps memory. If this is not implemented, there is a real danger that the file system will be corrupted. In particular, the super-block will not be updated prior to the crash dump.

g) In single user mode, the operator knows what he is doing. In this mode the update program is not started, and the system must be manually forced to perform a filesystem update via the sync command.

h) The maximum number of user tasks in the system is 13. This is determined by the swapspace available to the system. It is probably adequate for two terminals, where 9 user task can be run (in addition to init, update and the two shells running in the other four process slots).

i) The user space is 16K words (ie. 28K memory is available to the system). This requires a swap size of 66 blocks on the disk (calculated as 2 blocks + 4 x UCORG expressed in 1024 word blocks). As the system space is increased, the size of the swapsize can be reduced. This will allow more processes to be supported. Alternatively, the swap space can be increased by changing the swapdevice to allow more processes to be entered into the process table.

THE UNIVERSITY OF NEWCASTLE
NEW SOUTH WALES, 2308

COMPUTING CENTRE

TELEPHONE 68 0401
EXT. 305

23 February 1979

Ian Johnstone
AGSM
PO Box 1
Kensington NSW 2033

Would you please add my name to the AUUGN mailing list (including, if possible, the vol I nos I & II)

John Lambert

JOHN A LAMBERT
COMPUTING CENTRE
UNIVERSITY OF NEWCASTLE
NSW 2308

phone (049) 68-5305

The University of Wollongong
P.O.Box 1144 Wollongong N.S.W. 2500 Australia
Telephone (042) 297311 Telex 29022 Cable Uniofwol
In Reply Please Quote: JR.at

March 19, 1979

Mr Ian Johnstone,
Australian School of Management
University of New South Wales,
KENSINGTON, N.S.W. 2033

Dear Ian,

Enclosed is some information on EDUCOM for the next UNIX newsletter.

Yours sincerely,

Prof. J. Reinfelds,
Chairman,
Dept. of Computing Science

Enclosure

EDUCOM is a non-profit educational corporation founded in 1964 to promote the most effective use of computing and other technologies in higher education. Some 270 colleges and universities are members.In 1978 NSF awarded a grant of $362,000 to EDUCOM for a 20 month study aimed at identifying and understanding the major factors associated with computer-based resource sharing in academic research and teaching. Some of the principal activities of EDUCOM are:

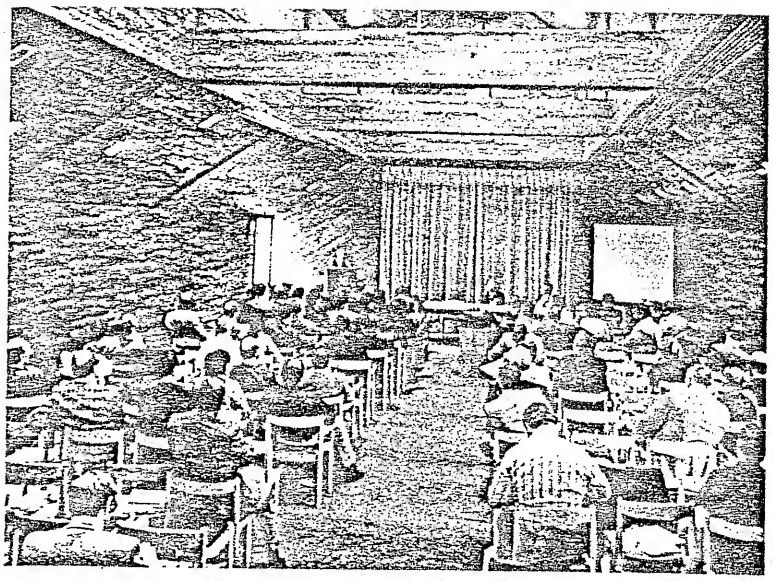
- Publishing a quarterly magazine, conference proceedings, research reports and more graphics.

- Providing specialized consulting services such as a review of an institution's current computing activities with possible recommendations for improvement.

- Conducting one-day workshop seminars on topics of special interest.

- Arrangement for member discounts with vendors of computers, terminals and software products.

EDUCOM membership is institutional and for Universities outside of North America the annual membership fee is $250.00.



WHAT IS EDUCOM?

EDUCOM is a nonprofit organization founded in 1964 to promote cooperative efforts and deal with common problems in the application of systems technology in higher education and research. Its activities are concerned with such areas as:

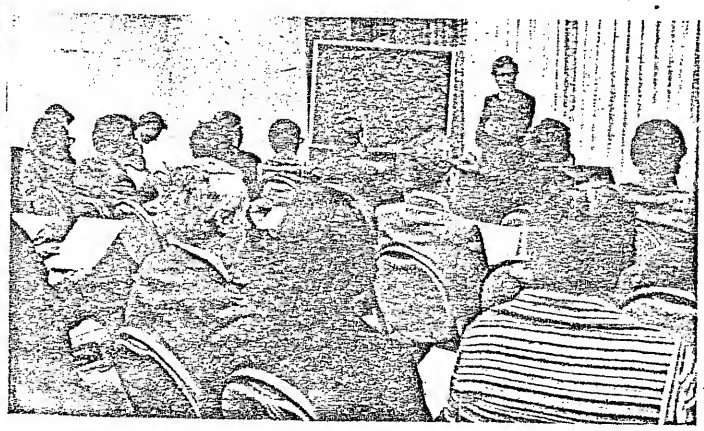
- the application and management of computer technology in academic instruction and research.
- the use of data processing and management information systems in college and university administration.
- the applications of computer technology in libraries and for information dissemination.

EDUCOM's current membership includes over 250 university and college campuses, which together enroll more than one-third of all students attending four-year colleges in the United States. Its unique contribution is to bring together faculty, administrators, and technical experts from member institutions to focus on effective and economical ways of using systems technology to satisfy the needs of higher education.

EDUCOM serves the higher educational community through activities such as:

- disseminating information and promoting cross-fertilization of ideas through conferences, workshops, and publications.
- promoting sharing and exchange of specialized computer and information resources among colleges and universities.
- providing consulting services on the management and use of computer technology in institutions of higher education.
- negotiating discount agreements with computer equipment suppliers for the benefit of members.
- promoting research on problem areas of general concern to the higher educational community.

EDUCOM is organized so that its member institutions can and do play an active role in developing goals and policies. Institutional Representatives, appointed by the President of each member institution, serve as the link between EDUCOM and the faculty and administration of their campuses. Representatives bring to EDUCOM conferences and task forces the views, problems, and proposals of their institutions concerning computer and communications technology, and share with their colleagues on the campus the latest ideas and developments obtained through EDUCOM.



WHAT ARE SOME OF THE ISSUES EDUCOM ADDRESSES?

Developing systems for networking and resource sharing.

The need for and the advisability of discipline-oriented computer centers.

Pricing and budgeting for computing services in colleges and universities.

The relative place of computing in institutional priorities.

Uses of minicomputers and hierarchical computing.

The interface between university administrators and university computing.

The potential educational use of cable television and its integration with computing.

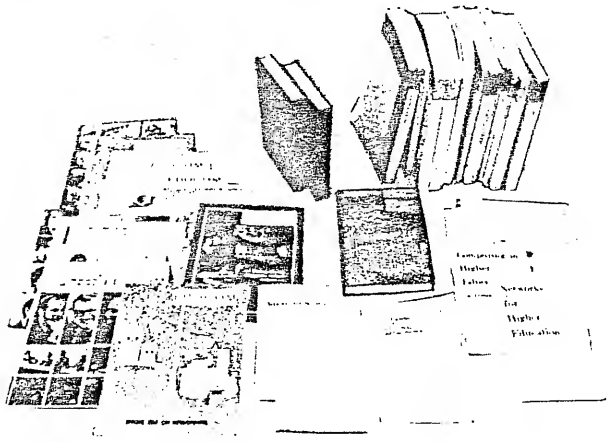
Satisfying the growing demand for computing capability in higher eduction.

The role of state governments in educational computing.

The role of simulation modeling in applying computing to university organization.

The use of computer and video technology in instruction.

Cost-effective ways of providing library and bibliographic services.



W

The EDUCOM Consulting Group provides analysis and advice relating to the planning and use of computer and other technologies in higher education. Drawing on experts from EDUCOM member institutions and members of the EDUCOM staff, a consulting team of individuals with appropriate skills and experience is formed for each assignment. Recently completed consultations have dealt with planning for computing activities, computing center organization, university management information systems, acquisition of minicomputer systems, and models for financial planning.

The EDUCOM Bulletin, a quarterly magazine, is in its twelfth year of publication. Circulated to over 10,000 faculty and staff members of EDUCOM institutions, the Bulletin provides reports on presentations at conferences, reports of research projects, descriptions of applications of technology to higher education, and articles by authorities in fields describing functioning systems of interest to the educational community.

Planning Council. The establishment of the Planning Council on Computing in Education and Research ushered in a new stage in EDUCOM's development. During 1975, its first full year of operation, the Planning Council developed a research plan and initiated a number of studies related to computer networks. Twenty-one colleges and universities participate in the Planning Council, and each provides financial support of $10,000 per year for five years. The Council is governed through a Policy Board and a Technical Committee; each member institution has a senior representative on each of these committees. Planning Council activities include the development of a prototype network for computing resource sharing, benchmark studies of members' computer centers, meetings of regional networking groups, support of discipline-oriented computing user groups, and a variety of special studies and projects leading to the enhancement of the network.

EDUNET. With encouragement from EDUCOM and funding from the Planning Council, a group of consumers and suppliers of university and college computing services began to share

EDUCOM conducts research on the application of computer and communications technology to areas in which significant results can be achieved with the combined resources of its members.

Past research has included studies of:

- The use of information systems in medicine and libraries.
- Plans for a biomedical communications network.
- System configuration and technology required for an agricultural information network.

HAT DOES EDUCOM PROVIDE FOR YOU?

Annual Conferences provide a forum for discussing resource sharing, computer networking, the development of information systems, and other topics of current interest.

Proceedings of each regular conference are published in paperback book form and distributed free to Institutional Representatives as well as to conference participants.

Special Seminars are scheduled on topics of concern to members. Funded by foundations or attendance fees, these meetings provide a source of expertise and experience to all members. Seminars are typically scheduled for one day, and, to minimize travel expense for participants, are repeated in regional locations if registration warrants.

Research Reports and Monographs are published in areas of interest to higher education. Published as paperback or hard cover books, these reports are available at reduced prices to faculty and staff of member institutions.

Discounts on the purchase or lease of computing equipment and related materials from selected vendors are available to EDUCOM members. These arrangements allow individual members to deal directly with the vendor, taking advantage of the centrally negotiated discount.

Informal Communications to members include letters to presidents, memos to Institutional Representatives, and other mailings that keep member institutions informed about developments in the sharing of resources and the use of computing and other technologies.

SPECIAL ACTIVITIES

resources through networks in 1976. The set of resources and facilitating services provided by EDUCOM is called EDUNET. The computing facilities at MIT, Dartmouth, SUNY, Yale, and Stanford are currently available through Telenet and/or TYMNET, both "value-added" communications vendors. EDUNET also includes several other university and research facilities that are not Planning Council members. Further additions are expected in the future. One of the facilitating services supplied to EDUNET participants is an on-line directory of the resources available, which is accessible through the SPIRES database management system at Stanford University. As additional computing resources participate in EDUNET, the on-line directory will be expanded.

Network Simulation Project. Closely related to Planning Council activities is a major EDUCOM research project that has developed a simulation model of an inter-institutional computer network. Use of the model in gaming exercises will help answer questions about networking and its potential impact on participating institutions.

EDUCOM Council Task Forces enable EDUCOM to draw on faculty and administration of member institutions to develop guidelines, checklists, and reports using the combined experience of members to deal with pressing current problems of a general nature.

Legal Education. Discounts have been arranged for EDUCOM members subscribing to a computer-based legal reference service. Experiments in using computer-based exercises for teaching law over a network are being conducted. The results of these experiments should have general applicability to other disciplines and will be disseminated through an EDUCOM report.

Financial Planning. Computer-based models for long-range financial planning developed at Stanford University are being adapted for use by other colleges and universities. The generalized models and procedures for using them will be made available to other interested institutions at the conclusion of the project.

RESEARCH PROJECTS

- Factors inhibiting the use of computers in instruction.
- The use of computer simulation in socio-economic policy research.
- Planning for computing service for higher education by state agencies and institutions in the United States and Canada.

Current research includes:

- Use of a simulation model in gaming exercises to gain a better understanding of institutional behavior in a computer network.
- Pricing and budgeting of computing services within colleges and universities.
- The provision of user services to aid users in accessing computing resources.
- A comparative study of the manner in which states have developed and implemented plans for providing computing service to higher education.